

NFSv4
Internet-Draft
Intended status: Standards Track
Expires: March 3, 2018

J. Fields
A. Gruenbacher
Red Hat
August 30, 2017

Allowing Inheritable NFSv4 Access Control Entries to Override the Umask [draft-ietf-nfsv4-umask-05](#)

Abstract

In many environments, inheritable NFSv4 Access Control Entries (ACEs) can be rendered ineffective by the application of the per-process umask. This can be addressed by transmitting the umask and create mode as separate pieces of data, allowing the server to make more intelligent decisions about the permissions to set on new files. This document proposes a protocol extension which accomplishes that.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of [BCP 78](#) and [BCP 79](#).

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on March 3, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to [BCP 78](#) and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in [Section 4.e](#) of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1.	Conventions Used in This Document	2
2.	Problem Statement	2
3.	Protocol Extension Considerations	3
4.	XDR Extraction	3
5.	mode_umask Attribute	4
6.	Security Considerations	5
7.	IANA Considerations	5
8.	References	5
8.1.	Normative References	5
8.2.	Informative References	6
Appendix A.	Acknowledgments	6
	Authors' Addresses	6

[1.](#) Conventions Used in This Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [\[RFC2119\]](#).

[2.](#) Problem Statement

On Unix-like systems, each process is associated with a file mode creation mask (umask), which specifies which permissions must be turned off when creating new file system objects.

When applying the mode, [section 6.4.1.1 of \[RFC7530\]](#) recommends that servers SHOULD restrict permissions granted to any user or group named in the Access Control List (ACL) to be no more than the permissions granted by the MODE4_RGRP, MODE4_WGRP, and MODE4_XGRP bits. Servers aiming to provide clients with Unix-like chmod behavior may also be motivated by the same requirements in [\[SUSv4\]](#). (See the discussion of additional and alternate access control mechanisms in section "4.4 File Permissions" of that document.)

On many existing installations, all ordinary users by default use the same effective group ID. To prevent granting all users full access to each other's files, such installations usually default to a umask with very restrictive permissions. As a result, inherited ACL entries (inheritable ACEs) describing the permissions to be granted to named users and groups are often ignored. This makes inheritable ACEs useless in some common cases.

Linux solves this problem on local filesystems by ignoring the umask in the case the parent of the newly-created file has inheritable ACEs; see [[LinuxACL](#)].

The same solution should work for NFS. However, the NFSv4 protocol does not currently give the client a way to transmit the umask of the process opening a file. And clients have no way of atomically checking for inheritable permissions and applying the umask only when necessary. As a result, the server receives an OPEN with a mode attribute that already has the umask applied.

This document solves the problem by defining a new attribute which allows the client to transmit umask and the mode specified at file creation separately, allowing the client to ignore the umask in the presence of inheritable ACEs. At least in the Linux case, this allows NFSv4 to provide the same semantics available using local access.

3. Protocol Extension Considerations

This document presents an extension to minor version 2 of the NFSv4 protocol as described in [[nfsv4-versioning](#)]. It describes a new OPTIONAL feature. NFSv4.2 servers and clients implemented without knowledge of this extension will continue to interoperate with clients and servers that are aware of the extension (whether they support it or not).

Note that [[RFC7862](#)] does not define NFSv4.2 as non-extensible, so that it is considered by [[nfsv4-versioning](#)] to be an extensible minor version. As a result, upon publication of this document as a Proposed Standard, the extension described herein will effectively be part of NFSv4.2, even though this document does not update [[RFC7862](#)] or [[RFC7863](#)].

4. XDR Extraction

The additional lines of external data representation (XDR) [[RFC4506](#)] description embedded in this document can be extracted by feeding this document into the following shell script:

```
<CODE BEGINS>
```

```
#!/bin/sh
grep '^ *////' $* | sed 's?^ *//// ??' | sed 's?^ *////$??'
```

```
<CODE ENDS>
```


That is, if the above script is stored in a file called "extract.sh", and this document is in a file called "umask.txt", then the reader can do:

```
sh extract.sh < umask.txt > umask.x
```

The effect of the script is to remove leading white space from each line, plus a sentinel sequence of "///".

Once that extraction is done, these added lines need to be inserted into an appropriate base XDR of the generated XDR from [\[RFC7863\]](#), together with XDR from any additional extensions to be recognized by the implementation. This will result in a ready-to-compile XDR file.

5. mode_umask Attribute

<CODE BEGINS>

```
/// struct mode_umask4 {
///     mode4  mu_mode;
///     mode4  mu_umask;
/// };
///
///  /*
///   * New For UMASK
///   */
/// const FATTR4_MODE_UMASK          = 81;
```

<CODE ENDS>

Name	Id	Data Type	Acc	Defined in
mode_umask	81	mode_umask4	W	Section 5

Table 1

The NFSv4.2 mode_umask attribute is based on the umask and on the mode bits specified at open time, which together determine the mode of a newly created UNIX file. Only the nine low-order mode4 bits of mu_umask are defined. A server MUST return NFS4ERR_INVALID if bits other than those nine are set.

The mode_umask attribute is only meaningful for operations that create objects (CREATE and OPEN); in other operations that take fattr4 arguments, the server MUST reject it with NFS4ERR_INVALID.

The server MUST return NFS4ERR_INVALID if the client attempts to set both mode and mode_umask in the same operation.

When the server supports the mode_umask attribute, a client creating a file should use mode_umask in place of mode, with mu_mode set to the unmodified mode provided by the user, and mu_umask set to the umask of the requesting process.

The server then uses mode_umask as follows:

- o On a server that supports ACL attributes, if an object inherits any ACEs from its parent directory, mu_mode SHOULD be used, and mu_umask ignored.
- o Otherwise, mu_umask MUST be used to limit the mode: all bits in the mode MUST be turned off which are set in the umask; the mode assigned to the new object becomes (mu_mode & ~mu_umask) instead.

6. Security Considerations

The mode_umask attribute shifts to the server the decision about when to apply the umask. Because the server MUST apply the umask if there are no inheritable permissions, the traditional semantics are preserved in the absence of a permission inheritance mechanism. The only relaxation of permissions comes in the case servers follow the recommendation that they ignore the umask in the presence of inheritable permissions.

The practice of ignoring the umask when there are inheritable permissions in the form of a "POSIX" default ACL is of long standing and has not given rise to security issues. The "POSIX" default ACL mechanism and the mechanism for permission inheritance in NFSv4 are equivalent from a security perspective.

7. IANA Considerations

This document does not require any actions by IANA.

8. References

8.1. Normative References

[nfsv4-versioning]

Noveck, D., "Rules for NFSv4 Extensions and Minor Versions", [draft-ietf-nfsv4-versioning-08](#) (work in progress), December 2016.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", March 1997.
- [RFC4506] Eisler, M., "XDR: External Data Representation Standard", STD 67, [RFC 4506](#), May 2006.
- [RFC7530] Haynes, T. and D. Noveck, "Network File System (NFS) version 4 Protocol", [RFC 7530](#), March 2015.
- [RFC7862] Haynes, T., "Network File System (NFS) Version 4 Minor Version 2 Protocol", [RFC 7862](#), November 2016.
- [RFC7863] Haynes, T., "Network File System (NFS) Version 4 Minor Version 2 External Data Representation Standard (XDR) Description", [RFC 7863](#), November 2016.

8.2. Informative References

- [LinuxACL] Gruenbacher, A., "ACL(5) - Access Control Lists", Linux man pages ACL(5), March 2002, <<http://kernel.org/doc/man-pages/online/pages/man5/acl.5.html>>.
- [SUSv4] The Open Group, "Single UNIX Specification Version 4", 2013.

Appendix A. Acknowledgments

Thanks to Trond Myklebust and Dave Noveck for the suggestion to define this as a (mode, umask) pair rather than just umask. Thanks for review to them and to Warren Kumari, Adam Roach, Spencer Dawkins, Mike Kupfer, and Thomas Haynes for review, and to Thomas Haynes for XDR help.

Authors' Addresses

J. Bruce Fields
Red Hat, Inc.

Email: bfields@redhat.com

Andreas Gruenbacher
Red Hat, Inc.

Email: agruenba@redhat.com

